APPENDIX DESCRIBING PROTOCOLS

This appendix provides details for the protocols described in Detailed Description. The protocol is initiated by a delegate p. In practice, more than one delegate could initiate the protocol for the same given request because an on-line CA server p starts acting as a delegate when p first receives the request or when p receives any message related to the processing of the request. The optimizations outlined in the Detailed Description are not included in this presentation.

The following notational conventions are used throughout the appendix:

- p, q: on-line CA servers
- c: on-line CA client
- $\langle m \rangle_k$: message m signed by on-line CA using its service private key k
- $\langle m \rangle_p$: message m signed by an on-line CA server p using p's private key
- $\langle m \rangle_c$: message m signed by a client c using c's private key
- $PS(m, s_p)$: a partial signature for a message m generated by an on-line CA server p using p's share s_p
- $[h_1 \rightarrow h_2 : m]$: message m is sent from host (an on-line CA server or a client) h_1 to host h_2
- $[\forall q.\ p \rightarrow q : m_q]$: message m_q is sent from on-line CA server p to on-line CA server q for every on-line CA server q

Each message includes a type identifier to indicate the purpose of the message.

Client Protocol.

Every client request has the form:

$\langle type, c, seq, parm, cred \rangle_c$,

where "type" indicates the type of the request, "c" is the client issuing the request, "seq" is a unique sequence number for the request, "parm" are parameters related to the request, and "cred" is credentials that authorize the request.

Clients use the following protocol to communicate with the on-line CA.

1. To invoke Query for the certificate associated with name cid, client c composes a request:

$$R = \langle query,\ c,\ seq,\ cid,\ cred \rangle_c$$

To invoke Update to establish a new binding of key with name cid based on a given certificate ζ' for cid, client c composes a request:

$$R = \langle update,\ c,\ seq,\ \zeta',\ \langle cid,\ key \rangle,\ cred \rangle_c$$

2. Client c sends R to t+1 on-line CA servers. It periodically re-sends R until it receives a response to its request. For a Query, the response will have the form $\langle R,\ \zeta \rangle_k$, where ζ is a certificate for cid. For an Update, the response will have the form $\langle R,\ done \rangle_k$.

Threshold Signature Protocol.

The following describes threshold signature protocol threshold_sign(m, E), where m is the message to be signed and E is the evidence used in self-verifying messages to convince receivers to generate partial signatures for m. While this protocol is appropriate for schemes such as threshold RSA, the protocol might not be applicable to other threshold signature schemes, such as those based on discrete logarithms. Those schemes may require an agreed-upon random number in generating partial signatures. Such schemes can be implemented by adding a new first step, in which a delegate decides a random number based on suggestions from t + 1 on-line CA servers (to ensure randomness) and notifies others of this random number, before on-line CA servers can generate partial signatures. Different evidence is used in the protocols for Query and Update.

1. On-line CA server p sends to each on-line CA server q a sign_request message with message m to be signed and evidence E.

$$[\forall q.\ p \rightarrow q : \langle sign\_request,\ p,\ m,\ E \rangle_p] \quad (i)$$

2. Each on-line CA server q, upon receiving a sign_request message (i), verifies evidence E with respect to m. If E is

valid, then q generates a partial signature using its share s_q and sends the partial signature back to p.

$$[q \rightarrow p : \langle \text{sign\_response}, q, p, m, PS(m, s_q)\rangle_q]$$

3. On-line CA server p periodically repeats step 1 until it receives partial signatures from a quorum of on-line CA servers (which includes a partial signature from p itself). In fact, p can try to construct the signature as soon as it has received t + 1 partial signatures. p has to wait for more partial signatures only if some partial signatures it received are incorrect. On-line CA server p then selects t + 1 partial signatures to construct signature $\langle m\rangle_k$. If the resulting signature is invalid (which would happen if compromised on-line CA servers submit erroneous partial signatures), then p tries another combination of t + 1 signatures. In the worst case, p must try t+1 out of a total of 2t+1 combinations. The cost is insignificant when t is small. There are conventional robust threshold cryptography schemes that can reduce the cost using error correction codes. This process continues until the correct signature $\langle m\rangle_k$ is obtained.

Query processing protocol.

1. Upon receiving a request $R = \langle \text{query}, c, seq, cid, cred\rangle_c$ from a client c, on-line CA server p first checks whether R is valid based on the credentials cred provided. If it is valid then p sends a query_request message to all on-line CA servers:

$$[\forall q.\ p \rightarrow q : \langle \text{query\_request}, p, R\rangle_p] \quad \text{(ii)}$$

2. Each on-line CA server q, upon receiving query request message (ii), checks the validity of the request. If the request is valid, then q fetches the current signed local certificate associated with name cid: $\zeta_q = \langle cid, \sigma(\zeta_q), key_q\rangle_k$. On-line CA server q then sends back to p the following message:

$$[q \rightarrow p : \langle \text{query\_response}, q, p, R, \zeta_q\rangle_q]$$

3. On-line CA server p repeats step 1 until it receives the query response messages from a quorum of on-line CA servers (including p itself). p verifies that the certificates in these

messages are correctly signed by the on-line CA. Let $\zeta = \langle cid, \sigma, key \rangle_k$ be the certificate with the largest serial number in these query response messages. On-line CA server p invokes threshold_sign(m, E), where m is (R, ζ) and E is the query response messages collected from a quorum of on-line CA servers, thereby obtaining $\langle R, \zeta \rangle_k$.

4. On-line CA server p sends the following response to client c:

$$[p \rightarrow c : \langle R, \zeta \rangle_k].$$

To implement the optimization described in the Detailed Description, p also forwards the response to all other on-line CA servers. Henceforth, these on-line CA servers do not need to act as a delegate for this request any more. The same is true for the last step of Update request processing.

Update processing protocol.

1. Upon receiving a request $R = \langle update, c, seq, \zeta', \langle cid, key \rangle, cred \rangle_c$ from a client c, on-line CA server p first checks whether R is valid, based on the credentials cred provided. If it is valid then p computes serial number $\sigma(\zeta) = (v + 1, h(R))$ for new certificate ζ, where v is the version number of ζ' and h is a public collision-free hash function, and invokes threshold_sign(m, E), where m is $\langle cid, \sigma(\zeta), key \rangle$ and E is R, thereby obtaining $\zeta = \langle cid, \sigma(\zeta), key \rangle_k$.

2. On-line CA server p then sends an update_request message to every on-line CA server q.

$$[\forall q.\ p \rightarrow q : \langle update\_request, p, R, \zeta \rangle_p] \quad (iii)$$

3. Each on-line CA server q, upon receiving an update_request message (iii), updates its certificate for cid with ζ if and only if $\sigma(\zeta_q) < \sigma(\zeta)$, where ζ_q is the certificate for cid stored by the on-line CA server. On-line CA server q then sends back to p the following message:

$$[q \rightarrow p : \langle update\_response, q, p, R, done \rangle_q]$$

4. On-line CA server p repeats step 2 until it receives the

update_response messages from a quorum of on-line CA servers. p then invokes threshold_sign(m, E), where m is (R, done) and E is the update_response messages collected from a quorum of on-line CA servers, thereby obtaining $\langle R, \text{done} \rangle_k$.

5. On-line CA server p sends the following response to client c:

$$[p \rightarrow c : \langle R, \text{done} \rangle_k]$$